京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

京都大学
KYOTO UNIVERSITY

Title	安定化理論に基づくlog methodについて (Computer Algebra : Design of Algorithms, Implementations and Applications)
Author(s)	白柳, 潔; 関川, 浩
Citation	数理解析研究所講究録 (2009), 1666: 98-105
Issue Date	2009-10
URL	http://hdl.handle.net/2433/141073
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

Kyoto University

数理解析研究所講究録
第1666巻 2009年 98-105

安定化理論に基づくlog methodについて

白柳 潔*
東海大学理学部
KIYOSHI SHIRAYANAGI
SCHOOL OF SCIENCE, TOKAI UNIVERSITY

関川 浩†
日本電信電話株式会社　NTT コミュニケーション科学基礎研究所
HIROSHI SEKIGAWA
NTT COMMUNICATION SCIENCE LABORATORIES, NIPPON TELEGRAPH AND TELEPHONE CORPORATION

1　はじめに

安定化理論 [10] は、近似計算で実行すると不安定となるアルゴリズムに対し、それを変形して近似計算で実行しても誤差の影響を抑制し、安定な出力が得られるようにするための理論である。本論では、その安定化理論の全く新しい応用として、なるべく正確計算を軽減させながらも最終的に正確係数の正しい出力を得るための手法を提案する。

[11, 12] において、我々は既に安定化理論に基づく log method を提案している。これは、アルゴリズム実行中の係数の log（記録）を取っておき、計算は浮動小数点計算に任せ、最終的に log から正確係数の出力を得るというアイデアである。本論で提案する新手法は、基本的にこのアイデアを用いるが、ゼロ書換えが真に正しいかどうかをその都度確認するもので、出力の正当性を確認する必要がないことが前者との大きな相違である。本論に関連の深い他の研究には、[5, 6, 2] などがある。

2節で、安定化手法と以前提案した log method である ISZ 法について復習する。3節で、新しい log method である ISCZ 法を提案し、計算機実験について報告する。

2　復習

2.1　安定化手法

次のアルゴリズムを対象に安定化理論の復習を簡単に行う。詳細は [9, 8, 10] を参照されたい。

- データは、すべて多項式環 $R[x_1, \ldots, x_m]$ の元からなる。R は実数体の部分体である。
- データ間の演算は、$R[x_1, \ldots, x_m]$ 内の加減乗または剰余計算である。

*shirayan@ss.u-tokai.ac.jp
†sekigawa@theory.brl.ntt.co.jp

- データ上の述語は、不連続点をもつとすればそれは 0 のみである。

述語の不連続点が 0 という意味は、If "$C = 0$" then ... else ... のように、値が 0 か否かによって分岐が別れることである。上記クラスのアルゴリズムを、不連続点 0 の代数的アルゴリズムとよぶ。ほとんどの数式処理のアルゴリズムはこのクラスに入るか、このクラスのアルゴリズムに変換可能である。

さて、安定化の 3 つのポイントは、

- アルゴリズムの構造は変えない。
- データ領域において、ふつうの係数を区間係数に変える。
- 述語の評価の直前で、区間係数のゼロ書換えを行なう。

である。すなわち、安定化されたアルゴリズムは次のようになる。

区間領域 データ領域は区間係数多項式の集合。区間係数は $[A, \alpha]$ なる形で、$A \in R$, α は非負の実数。$[A, \alpha]$ は集合 $\{x \in R \mid |x - A| \leq \alpha\}$ を意味する。

区間演算 二項演算 $* \in \{+, -, \times, /\}$ に対し、

$$[A, \alpha] * [B, \beta] = [A * B, \gamma_*].$$

ここに、γ_* は次を満たす。

$$|x - A| \leq \alpha, |y - B| \leq \beta \Rightarrow |x * y - A * B| \leq \gamma_*.$$

ゼロ書換え 不連続点 0 をもつ述語を評価する直前で、各区間係数 $[C, \gamma]$ に対し、

$|C| \leq \gamma$ ならば $[C, \gamma]$ を [0,0] に書き換えよ。

$|C| > \gamma$ ならばそのままとせよ。

区間演算の具体的な定義については、文献 [1] を参照されたい。

今、入力 $f \in R[x_1, \ldots, x_m]$ を

$$f = \sum_{i_1, \ldots, i_m} a_{i_1 \ldots i_m} x_1^{i_1} \cdots x_m^{i_m}$$

と表したとき、f に対する近似列 $\{Int(f)_j\}_j$ を

$$Int(f)_j = \sum_{i_1, \ldots, i_m} [(a_{i_1 \ldots i_m})_j, (\alpha_{i_1 \ldots i_m})_j] \, x_1^{i_1} \cdots x_m^{i_m}$$

で定義する。ここに、すべての $i_1, \ldots, i_m$ について、

$$|a_{i_1 \ldots i_m} - (a_{i_1 \ldots i_m})_j| \leq (\alpha_{i_1 \ldots i_m})_j \text{ for } \forall j$$

$$(\alpha_{i_1 \ldots i_m})_j \to 0 \text{ as } j \to \infty$$

このとき、単に

$$Int(f)_j \to f$$

と書く。

さて、$\mathcal{A}$ を安定化したアルゴリズムを $Stab(\mathcal{A})$ と書くと、次が安定化理論の基本定理である。

定理 1 (安定化理論の基本定理) $\mathcal{A}$ は不連続点 0 の代数的アルゴリズムで、入力 $f \in R[x_1, \ldots, x_m]$ に対し正常終了するとせよ。このとき、f に対する任意の近似列 $\{Int(f)_j\}_j$ に対し、ある n が存在して、$j \geq n$ ならば、$Stab(\mathcal{A})$ は $Int(f)_j$ に対し正常終了し、

$$Stab(\mathcal{A})(Int(f)_j) \to \mathcal{A}(f).$$

簡明を期すため、入力は一つだけの多項式にしているが、入力はもちろん、多項式の有限集合でもよい。主題のグレブナ基底の場合も、入力は多項式の集合である。本定理の Buchberger のアルゴリズムに特化した証明は [7] に、より一般的な場合に対する厳密な証明は [10] にある。

2.2 ISZ 法

2.2.1 シンボル付き区間

区間と形式的なシンボルを組み合わせた係数（シンボル付き区間）を導入する。これは、後に 3 節で提案する手法においても使われる。区間は、従来と同様の意味の区間である。シンボルは、アルゴリズム実行中に現れる係数の log（記録）を取るのに使われる。例えば、入力係数が 1/3 と 1/9 だったとしよう。これらの精度 3 の（円形）区間はそれぞれ $[.333, .0005]$ と $[.111, .0005]$ である。さて、1/3 に対するシンボルを s、1/9 に対するシンボルを t として、区間と組み合わせると、それぞれ、$[[.333, .0005], s]$ と $[[.111, .0005], t]$ となる。次に、これらの間の演算、例えば、加算を、

$$[[.333, .0005], s] + [[.111, .0005], t] = [[.333, .0005] + [.111, .0005], \dot{+}(s, t)]$$

と定義する。$[.333, .0005] + [.111, .0005]$ に対しては通常の区間演算を使う。シンボル部分 $\dot{+}(s, t)$ は再び形式的なシンボルで、加算を実施したことを記録できれば何でもよい。効率的なシンボル付けについては 2.2.3 節で議論する。

アルゴリズム終了後、最終的なシンボルを正確係数に復元する。先の簡単な例で言えば、もし最終的なシンボルが $\dot{+}(s, t)$ であったとすれば、s に 1/3 を、t に 1/9 を代入し、$\dot{+}$ には加算の意味を与えて、$1/3+1/9 = 4/9$ と復元する。

シンボル付き区間 (interval with symbol) のことを単に IS と呼ぶこともある。

2.2.2 手続き

$\mathcal{A}$ を不連続点 0 の代数的アルゴリズムとする。ISZ 法の手続きは次の通りである。

R-to-IS 各入力係数 a をペア $[[\tilde{a}, \alpha], Symbol_a]$ に変換する。ここに、$[\tilde{a}, \alpha]$ は a の予め定められた精度の区間、$Symbol_a$ は a を表すシンボルである。

IS 演算 IS 間の演算を次のように実行する：

$$[[A, \alpha], s] + [[B, \beta], t] = [[A, \alpha] + [B, \beta], \dot{+}(s, t)]$$

$$[[A, \alpha], s] - [[B, \beta], t] = [[A, \alpha] - [B, \beta], \dot{-}(s, t)]$$

$$[[A, \alpha], s] \times [[B, \beta], t] = [[A, \alpha] \times [B, \beta], \dot{\times}(s, t)]$$

すなわち、区間部分については区間演算を用い、シンボル部分については加算、減算、乗算の形式的なシンボル $\dot{+}$, $\dot{-}$, $\dot{\times}$ を使って、どういう演算が行なわれたかを記録する。

強制的ゼロ書換え 任意のIS $[[E,\epsilon],s]$ に対し、$|E| \leq \epsilon$ ならば、s が何であれ、$[[E,\epsilon],s]$ を $\mathbf{0}$ に書き換えよ。ここに、$\mathbf{0}$ は "ゼロIS" で、$\mathbf{0}t = 0$ for all $t \in R[x_1,\ldots,x_m]$.

IS-to-R 出力のシンボル部分の中の各入力シンボルにそれぞれ対応する入力係数を代入し、演算シンボルに演算の意味を与えて実数値に復元する。

正当性検証 実数係数に復元された出力が真に正しい出力かどうかをある手法によって確認する。YESならば、それを返し、NOならば、精度を上げて **R-to-IS** に戻る。ここに、ある手法とは "簡単に" 確認できる手法のことで、これが存在すると仮定する。

この手法をISZ法 (IS method with zero rewriting) と呼ぶ。これについて、次が成立する。

定理 2 (ISZ法の停止性と正当性) $\mathcal{A}$ が入力 I で正常終了するとせよ。このとき、$\mathcal{A}$ に対する *ISZ* 法は、常に有限ステップで終了し、正しい結果、すなわち、$\mathcal{A}(I)$ の出力と同じ結果を与える。

2.2.3 シンボルリスト

ISZ法は明らかにISのシンボル部分の膨張を招く。ここでは、それを防ぐための一つの工夫について述べる。

基本は、シンボルの代わりに数字を用いることである。すなわち、ISの代わりにペア $[[A,\alpha],M]$ を用いる。ここに、$[A,\alpha]$ は区間、M は入力シンボルまたは整数である。さらに、各ペアが他のペアからどのように構成されたかを示すリストを用意する。これをシンボルリストと呼ぶ。アルゴリズム終了後、このシンボルリストからISを復元する。

1. 2.2.2節で説明した **R-to-IS** 変換を行なう。
2. (初期化) N を0とし、シンボルリスト $\mathcal{L}$ を空リストとする。
3. ペア $[[A,\alpha],L]$ と $[[B,\beta],M]$ の間の演算 $*$ ($+$, $-$, $\times$ etc.) を次のように定義する。
   まず、区間部分を計算し、その結果の区間 $[C,\gamma] = [A,\alpha] * [B,\beta]$ にゼロ書換えを施す。
   もし、$[C,\gamma]$ が $[0,0]$ に書き換えられたら, その結果は $\mathbf{0}$ で、$\mathcal{L}$ はそのままとする。
   そうでなければ、N を $N+1$ とし、新しい係数 $[[C,\gamma],N]$ を作って、$\mathcal{L}$ の最後に $(*,L,M)$ を付加する。
4. (復元) $[[A,\alpha],N]$ が出力のある係数であるとする。$[[A,\alpha],N]$ を次のようにISに変換する：
   N がシンボルであれば、何もしない。$[[A,\alpha],N]$ がそのISそのものである。
   N が整数であれば、N を $\mathcal{L}$ の N 番目の要素に置き換え、以下再帰的に入力シンボルに行き着くまでこれを行なう。最終的に得られたペア $[[A,\alpha],S]$ がそのISである。

3 ISCZ法

3.1 理論

不連続点0の代数的アルゴリズムに対し、正確係数の出力を得るために正確計算を軽減するための新しい手法を提案する。基本的なアイデアは、ISZ法と同様、係数を記録するためにIS係数を使うが、各ゼロ

書換えにおいて、シンボルを復元して真にゼロ書換えが正しいかどうかを確認する。他方、ISZ 法では、ゼロ書換えが正しいかどうかに構わず、最後までアルゴリズムを実行するので、その点が決定的な違いである。本手法の手続きは以下の通りである：

R-to-IS 各入力係数 a をペア $[[\tilde{a},\alpha],Symbol_a]$ に変換する。ここに、$[\tilde{a},\alpha]$ は a の予め定められた精度の区間、$Symbol_a$ は a を表すシンボルである。

IS 演算 IS 間の演算を次のように実行する：

$$[[A,\alpha],s]+[[B,\beta],t]=[[A,\alpha]+[B,\beta],\dot{+}(s,t)]$$

$$[[A,\alpha],s]-[[B,\beta],t]=[[A,\alpha]-[B,\beta],\dot{-}(s,t)]$$

$$[[A,\alpha],s]\times[[B,\beta],t]=[[A,\alpha]\times[B,\beta],\dot{\times}(s,t)]$$

保証されたゼロ書換え 任意の IS $[[E,\epsilon],s]$ に対し、$|E|\leq\epsilon$ ならば、s をそれに対応する実数 $r(s)$ に復元する。もし、$r(s)=0$ ならば、次のステップに進む。そうでなければ、精度を上げて **R-to-IS** に戻る。

IS-to-R 出力のシンボル部分の中の各入力シンボルにそれぞれ対応する入力係数を代入し、演算シンボルに演算の意味を与えて実数値に復元する。

この手法を ISCZ 法 (IS method with *correct* zero rewriting) と呼ぶ。効率化のために、**保証されたゼロ書換え**において、後の再利用のために s の実数値 $r(s)$ を記憶しておくのも一法である。

さて、ある精度 j で $Int(f)_j$ を $Stab(\mathcal{A})$ に入力したときの実行過程は、もしアルゴリズム中のすべてのゼロ書換えが正しいならば、真の出力 f を $\mathcal{A}$ に入力したときの実行過程と完全に一致する。ISCZ 法では、各ゼロ書換えにおいて、それが正しいかどうかを確認する。さらに、安定化定理により、すべてのゼロ書換えが正しくなる精度が存在する。従って、ISCZ 法は有限ステップで終了し、その出力の各 IS 係数のシンボルは正しい正確係数を与える。

これを定理にまとめる。

定理 3 (ISCZ 法の停止性と正当性) $\mathcal{A}$ が入力 I で正常終了するとせよ。このとき、$\mathcal{A}$ に対する *ISCZ* 法は、常に有限ステップで終了し、正しい結果、すなわち、$\mathcal{A}(I)$ の出力と同じ結果を与える。

ISCZ 法の利点は、ISZ 法と違い、出力の正当性を確認する必要がないことである。任意の IS $[[E,\epsilon],s]$ に対し、$|E|\leq\epsilon$ でない限り、正確計算をスキップすることができ、浮動小数点計算だけで済む。換言すれば、ゼロでない係数についての正確計算を省略することができる。従って、本手法は、$|E|>\epsilon$ の場合が $|E|\leq\epsilon$ の場合よりも多いときに有効であるということができる。

3.2 計算機実験

ISCZ 法をグレブナ基底を計算する Buchberger のアルゴリズムに適用した。このアルゴリズムは、不連続点 0 の代数的アルゴリズムの典型的な例である。不連続点の 0 は、S 多項式の正規形がゼロかどうかをチェックするところだけではなく、多項式の主項あるいは主係数を取る操作にも陰に存在している。主項は、**ゼロでない**係数をもつ最大の項と定義されるからである。

次の 10 個の例題について計算機実験した。

1. $F=\{f_1,f_2,f_3\}$, ここに $f_1=\frac{1}{7}x^2-\frac{326548390854652}{272974017239}x+\frac{1263781236281}{712638126}y^2+\frac{26872672361827}{7263188218281}z^2$, $f_2=\frac{3}{8}xy+\frac{12367812638123}{763812368213132}yz-\frac{63812638126}{77263812831}y$, $f_3=\frac{4}{9}x+\frac{327091270979304}{24122375460421}y+\frac{18467031595309203}{318405459032}z-\frac{356318063693141319}{6436561806418109}$.

2. $F = \{(\sqrt{2}+\sqrt{5})x^3y + \sqrt{3}xy + \sqrt{7}, (\sqrt{3}-\sqrt{2})x^2y^2 - \sqrt{7}xy + 1/\sqrt{11}\}$.
3. $F = \{(\sqrt{2}+\sqrt{5})x^3y + \sqrt{3}xy + \sqrt{7}, (\sqrt{3}-\sqrt{2})x^2y^2 - \sqrt{7}xy + 1/\sqrt{5}\}$.
4. $F = \{ex + \sqrt{2}y + \sqrt{3}z, exy + \sqrt{5}yz + \sqrt{3}zx, xyz - e\}$, ここに、$e$ は Napier's number (2.71828...).
5. $F = \{\sqrt{2}ex^2 + xy^2 - z + 1/4, \sqrt{3}x + y^2z + 1/2, \sqrt{5}ex^2z - 1/2\,x - y^2\}$.
6. $F = \{\sqrt{2}ex^3y + \sqrt{3}xy + \sqrt{7}/e, \sqrt{3}/\pi \cdot x^2y^2 - \sqrt{7}xy + e/\sqrt{11}\}$.
7. $F = \{(\sqrt{2}+\sqrt{3})x^{30} + \sqrt{5} - 1, \sqrt{7}xy + \sqrt{11} + \sqrt{13}\}$.
8. $F = \{(\sqrt{2}+\sqrt{3})x^{21} + \sqrt{5} - 1, \sqrt{7}x^2y + \sqrt{11} + \sqrt{13}\}$.
9. $F = \{(\sqrt{2}+\sqrt{3})x^{31} + \sqrt{5} - 1, \sqrt{7}x^2y + \sqrt{11} + \sqrt{13}\}$.
10. $F = \{\sqrt{2}x^3 + \sqrt{3}y^3 + \sqrt{5}z^3 + \sqrt{7}, \sqrt{2}x^2 + \sqrt{3}y^2 + \sqrt{5}z^2 + \sqrt{11}, \sqrt{2}x + \sqrt{3}y + \sqrt{5}z + \sqrt{13}\}$.

例 1 は文献 [7] からの例、例 2, 3, 6 は同文献の他の例の変形である。例 4 は cyclic3 の変形、例 5 は文献 [3] の例の変形である。

すべての例 F について、我々はイデアル $\langle F\rangle$ の辞書式順序に関するグレブナ基底を計算した。計算機は、dual core AMD(R) Opteron(R) processor (2.85GHz), 8GB RAM, Linux(R) OS である。

我々は、次の 5 種類の方法を試みた。

1. Maple 12 で実装した ISCZ 法。初期精度は 1 で、精度の増加幅も 1。
2. Maple 12 で実装した ISCZ 法。初期精度は 10 で、精度は倍々で増加。
3. Maple 12 で自前で Buchberger のアルゴリズムを実装したもの (R_GB)。
4. Maple 12 の Groebner パッケージの組込関数 "Basis" (Maple GB)。
5. Singular 3-0-4 の組込関数 "groebner" (Singular GB)。

方法 1 と方法 2 の実験結果は、表 1 と表 2 に示す。方法 3,4,5 の実行 cpu 時間は、秒単位で表 3 に示す。

表 1,2 において、"# of skipped nonzero coefficients" は、方法 1,2 を使って成功に至った精度で、"$|E| > \epsilon$" であるゆえに正確計算がスキップされたゼロでない係数の個数を表す。表 3 において、"< 0.1" は計算が 0.1 秒以内に終わって結果が出たことを示す。"> n" (e.g. "> 3000") は、n 秒待っても計算は終わらず、結果が得られなかったことを示す。"-" は、Singular が超越数を受け付けないため実験ができなかったことを示す。"Segmentation fault" は計算が segmentation fault のため、終了したことを示す。

実験結果より、ISCZ 法は、有理係数のときよりも無理係数のときの方が有効であることがわかる。特に、超越数を含む場合はそれが顕著である。従って、一般に、有理係数の場合は FGLM アルゴリズム [4] やそのモジュラー版などを使えば事足りると思うが、超越数を含む無理係数の場合は、浮動小数点計算が効力を発揮するので、ISCZ 法を試してみるのも一法であろう。

4 おわりに

不連続点 0 の代数的アルゴリズムに対し、途中の計算の大部分を浮動小数点計算に任せつつも最終的には正確係数の結果を得るための手法を提案した。グレブナ基底計算のように、ゼロの係数よりもゼロでない係数が多く現れる場合、さらに係数が無理数の場合は、本手法は有効である。

今後の課題として、シンボルの格納や復元のためのより効率的な方法の開発や ISCZ 法をグレブナ基底計算以外の問題にも適用することなどが挙げられる。

表 1: 方法 1 の実験結果

Example	cpu time	initial precision	successful precision	# of zero rewriting	# of skipped nonzero coefficients
1	0.2	1	7	12	349
2	28.7	1	10	18	304
3	14.7	1	10	18	304
4	2.4	1	3	6	81
5	4008.2	1	12	43	1556
6	329.6	1	8	18	304
7	355.6	1	1	59	1978
8	69.9	1	1	21	420
9	147.1	1	1	31	850
10	71.0	1	10	24	903

表 2: 方法 2 の実験結果

Example	cpu time	initial precision	successful precision	# of zero rewriting	# of skipped nonzero coefficients
1	0.1	10	10	12	349
2	25.4	10	10	18	304
3	12.1	10	10	18	304
4	1.2	10	10	6	81
5	3142.9	10	20	43	1556
6	208.4	10	10	18	304
7	131.8	10	10	59	1978
8	25.6	10	10	21	420
9	61.2	10	10	31	850
10	55.7	10	10	24	903

表 3: 方法 3, 4, 5 の実行 CPU 時間

Example	R_GB	Maple GB	Singular GB
1	< 0.1	< 0.1	< 0.1
2	14.0	2.1	Segmentation fault
3	6.1	0.2	0.1
4	0.5	0.2	-
5	491.3	40.4	-
6	15.0	> 10000	-
7	1012.6	195.9	Segmentation fault
8	25.9	141.7	Segmentation fault
9	67.0	203.4	Segmentation fault
10	21.8	61.5	Segmentation fault

参 考 文 献

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations, Computer Science and Applied Mathematics.* Academic Press, 1983.

[2] M. Benouamer, D. Michelucci, and B. Peroche. Error-free boundary evaluation using lazy rational arithmetic: a detailed implementation. In *Proc. 2nd Symposium on Solid Modeling and Applications*, pages 115–126, 1993.

[3] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. Chapter 6 in N. K. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. D. Reidel Publishing Company, 1985.

[4] J. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Computation*, 16(4):329–344, Oct. 1993.

[5] J. R. Johnson. Real algebraic number computation using interval arithmetic. In *Proc. ISSAC 1992*, pages 195–205, 1992.

[6] J. R. Johnson and W. Krandick. Polynomial real root isolation using approximate arithmetic. In *Proc. ISSAC 1997*, pages 225–232, 1997.

[7] K. Shirayanagi. Floating point Gröbner bases. *Mathematics and Computers in Simulation*, 42(4-6):509–528, 1996.

[8] 白柳 潔. アルゴリズムの安定化理論. 数式処理, 5(2):2–21, 1997.

[9] 白柳 潔. 不安定なアルゴリズムを安定化する. 情報処理 39(2):111–115, 1998.

[10] K. Shirayanagi and M. Sweedler. A theory of stabilizing algebraic algorithms. Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 1995. 92 pages. (http://www.ss.u-tokai.ac.jp/~shirayan/msitr95-28.pdf)

[11] K. Shirayanagi and M. Sweedler. Automatic algorithm stabilization. In *ISSAC'96 poster session abstracts*, pages 75–78, 1996.

[12] K. Shirayanagi and M. Sweedler. Remarks on automatic algorithm stabilization. *J. Symbolic Computation*, 26(6):761–766, 1998.